Zadania – Python
środa, 22 listopada 2023

Zad. 1. Stwórz skrypt `zadanie1.py`:

1. Zdefiniuj funkcję gauss z jednym argumentem x, która będzie zwracać wartość standardowego rozkładu normalnego w punkcie x.

$$gauss(x) = \frac{1}{\sqrt{2\pi}} \exp\left(-\frac{x^2}{2}\right)$$

2. Narysuj wykres tej funkcji. Oznacz osie x i y (x, gauss(x)).
3. Wycałkuj tę funkcję metodą prostokątów w przedziale od -10 do 10 (Stwórz do tego ogólną funkcję przyjmującą xmin, xmax i n jako parametry):
   (a) Podziel przedział na n równoodległych punktów (na przykład x = np.linspace(xmin, xmax, n)).
   (b) Wylicz odległość dx między punktami (pamiętaj, że dla n punktów jest n-1 przedziałów).
   (c) Policz sumę wszystkich gauss(x) * dx w tym przedziale.
4. Narysuj wykres zależności całki powyżej od liczby punktów w przedziale n od 2 do 20. Narysuj 2 krzywe – jedną dla n parzystych, drugą dla nieparzystych. Obie krzywe powinny być na tym samym wykresie. Czy suma w obu przypadkach zbiega tak samo do 1?
5. Opisz osie x i y (n, P(-10<x<10)), oraz dodaj legendę do krzywych.
6. Do wykresu gaussa dorzuć krzywą dystrybuanty (tzn. funkcję której wartością jest całka oznaczona od -10 do x). Użyj funkcji zdefiniowanej wcześniej. Dodaj legendę.
7. Umieść wykres gaussa/dystrybuanty oraz wykres porównawczy całki w jednym oknie jako sąsiadujące wykresy. Użyj do tego polecenia plt.subplots.

Zad. 2. Stwórz skrypt `zadanie2.py`: narysuj histogramy pokazujące gęstość prawdopodobieństwa wyrzucenia określonej sumy przy użyciu n kości sześciennych. Użyj do tego plt.subplots z 2 kolumnami i 2 wierszami (4 wykresy dla liczby kostek n od 1 do 4). Dla każdego histogramu:

1. Wygeneruj dużą (np. 1000) ilość prób poprzez wylosowanie wartości sumy 1,2,3 lub 4 kostek (moduł numpy.random).
2. Wynik każdej próby umieść na histogramie. Opisz osie.
3. Dostosuj histogram tak, żeby jego biny odpowiadały możliwym sumom (czyli dla 1 kostki binów będzie 6 (od 1 do 6), dla 2 kostek binów będzie 11 (od 2 do 12), itd.).
4. Histogramy powinny pokazywać gęstość prawdopodobieństwa na osi y.

Zad. 3. Stwórz skrypt `zadanie3.py`: wczytaj dane (x, y, niepewność y) z pliku points.dat oraz:

1. Dopasuj do punktów funkcję, która według Ciebie najlepiej opisuje zależność (używając curve_fit z modułu scipy.optimize).
2. Podczas dopasowania ustaw początkową wartość parametrów.
3. Narysuj na jednym wykresie punkty (niepołączone ze sobą!) ze słupkami błędów, oraz dopasowaną funkcję.
4. Wypisz na ekran wartości dopasowanych parametrów oraz ich niepewności (można je uzyskać wyciągając pierwiastek z diagonali macierzy kowariancji, przykładowo np.sqrt(np.diag(cov))).
5. Opisz osie i dodaj legendę.
6. Zapisz rysunek do pliku fit.pdf.